IN THE CLAIMS

Please amend the claims as follows. Added text is underlined and deleted text is either struck through or shown in double enclosing brackets. Applicant avers that no new matter has been added.

1. (Currently Amended) A method, comprising:

in an object oriented run-time environment, after a classfile has been loaded:

a) invoking a second method from a first method instrumented with bytecode, said first method belonging to said classfile, said invoking comprising providing an identification of said first method and said classfile, said second method belonging to a dispatcher, said classfile having previously registered with said dispatcher, said dispatcher incapable of injecting bytecode into said classfile;

b) said dispatcher performing said second method to identify a plug-in module for said first method based upon said identification, said plug-in module to implement a handler method, said dispatcher returning to said first method an identifier of said plug-in module, said identifier not being a boolean value representative of one of only two possible states;

c) using said identifier to directly invoke said plug-in module to execute said handler method to report and/or record information about said first method;

d) executing said first method from a point beyond where said second method was invoked;

e) flowing from said first method to a third method;

f) invoking said second method from said third method, said invoking including providing an identification of said third method and a second classfile that said third method is a part of, said second classfile having been loaded and said third method instrumented with bytecode at least by the completion of e) above, said second classfile having previously registered with said dispatcher, said dispatcher incapable of injecting bytecode into said second classfile;

g) said dispatcher performing said second method to identify said plug-in module for said third method based upon said third method and second classfile identification, said dispatcher returning to said third method said identifier of said plug-in module;

h) using said identifier to directly invoke said plug-in module from said third method to execute said handler method to report and/or record information about said third method; and,

i) executing a portion of said third method from a point beyond where said second method was invoked from said third method.

2. (Previously Presented) The method of claim 1 wherein said executing of said handler method at c) above causes an entry time for said first method to be recorded.

3. (Previously Presented) The method of claim 1 wherein said executing of said handler method at c) above causes an exit time for said first method to be recorded.

4. (Previously Presented) The method of claim 1 wherein said executing of said handler method at c) above causes a counter maintained for said first method to be incremented.

5. (Previously Presented) The method of claim 1 wherein said executing of said handler method at c) above causes an input parameter value of said first method to be recorded.

6. (Previously Presented) The method of claim 1 wherein said executing of said handler method at c) above causes a returned value of said first method to be recorded.

7. (Original) The method of claim 1 wherein said first method is a constructor.

8. (Previously Presented) The method of claim 1 further comprising creating, prior to said invoking at a) above, an object having an input parameter value of said first method.

9. (Previously Presented) The method of claim 1 wherein said invoking at a) above further comprises providing an input parameter value of said first method.

10. (Previously Presented) The method of claim 1 wherein said invoking at a) above further comprises identifying where said first method's instructions can be found in memory.

11. (Previously Presented) The method of claim 1 further comprising, after said executing said first method from a point beyond where said second method was invoked but before said flowing to said third method at e) above:
invoking a fourth method from said first method because said first method is about to reach an exit point, said second method having been invoked from said first method because an entry point of said first method had just been reached, said dispatcher having said fourth method;
re-identifying said plug-in module for said first method as a consequence of said invoking a fourth method;
re-executing said handler method to report and/or record information about said first method; and,
executing a remaining portion of said first method through said exit point.

12. (Canceled)

13. (Previously Presented) The method of claim 1 wherein g) further comprises also identifying a second plug-in module for said third method based upon said third method and second classfile identification, said second plug-in module containing a second handler method.

14. (Previously Presented) The method of class 13 further comprising also executing said second handler method to report and/or record different information about said third method than what said first handler method reports and/or records about said third method.

15. (Previously Presented) The method of claim 14 wherein a first object is called to execute said first method and a second object is called to execute said fourth method.

16. (Previously Presented) The method of claim 15 wherein said object oriented run-time environment is a Java object oriented environment.

17. (Previously Presented) The method of claim 1 wherein said invoking at a) above further comprises providing said first method's signature, said first method's signature comprising:
said identification of said first method;
said identification of said classfile that said first method is a part of; and,
said first method's arguments.

18. (Currently Amended) An article of manufacture having stored thereon executable or interpretable program code which when processed by one or more computing systems cause a method to be performed, said method, comprising:
in an object oriented run-time environment, after a classfile has been loaded:
a) invoking a second method from a first method, said first method belonging to said classfile, said invoking comprising providing an identification of said first method and said classfile, said second method belonging to a dispatcher, said classfile having previously registered with said dispatcher<u>, said dispatcher being incapable of injecting bytecode into said classfile</u>;
b) said dispatcher performing said second method to identify a plug-in module for said first method based upon said identification, said plug-in module to implement a handler method, said dispatcher returning to said first method an identifier of said plug-in <u>module</u>, said identifier not being a boolean value that represents one of only two possible states;
c) using said identifier to directly invoke said plug-in module to execute said handler method to report and/or record information about said first method;
d) executing said first method from a point beyond where said second method was invoked;
e) flowing from said first method to a third method;

f) invoking said second method from said third method, said invoking including providing an identification of said third method and a second classfile that said third method is a part of, said second classfile having been loaded at least by the completion of e) above, said second classfile having previously registered with said dispatcher<u>, said dispatcher being incapable of injecting bytecode into said second classfile</u>;

g) said dispatcher performing said second method to identify said plug-in module for said third method based upon said third method and second classfile identification, said dispatcher returning to said third method said identifier of said plug-in <u>module</u>;

h) using said identifier to directly invoke said plug-in module to execute said handler method to report and/or record information about said third method; and,

i) executing a portion of said third method from a point beyond where said second method was invoked from said third method
wherein said dispatcher includes: i) a dictionary that is used to correlate said first and third methods to said plug-in module, and, ii) a plug-in pattern which lists said first and third methods along with additional methods that said dispatcher dispatches to said plug-in module.

19. (Previously Presented) The article of manufacture of claim 18 wherein said executing of said handler method at c) above causes an entry time for said first method to be recorded.

20. (Previously Presented) The article of manufacture of claim 18 wherein said executing of said handler method at c) above causes an exit time for said first method to be recorded.

21. (Previously Presented) The article of manufacture of claim 18 wherein said executing of said handler method at c) above causes a counter maintained for said first method to be incremented.

22. (Previously Presented) The article of manufacture of claim 18 wherein said executing of said handler method at c) above causes an input parameter value of said first method to be recorded.

23. (Previously Presented) The article of manufacture of claim 18 wherein said executing of said handler method at c) above causes a returned value of said first method to be recorded.

24. (Previously Presented) The article of manufacture of claim 18 wherein said first method is a constructor.

25. (Previously Presented) The article of manufacture of claim 18 further comprising creating, prior to said invoking at a) above, an object having an input parameter value of said first method..

26. (Previously Presented) The article of manufacture of claim 18 wherein said invoking at a) above further comprises providing an input parameter value of said first method.

27. (Previously Presented) The article of manufacture of claim 18 wherein said invoking at a) above further comprises identifying where said first method's instructions can be found in memory.

28. (Previously Presented) The article of manufacture of claim 18 further comprising, after said executing said first method from a point beyond where said second method was invoked but before said flowing to said third method at e) above:
invoking a fourth method from said first method because said first method is about to reach an exit point', said second method having been invoked from said first method because an entry point of said first method had just been reached, said dispatcher having said fourth method;
re-identifying said plug-in module for said first method as a consequence of said invoking a fourth method;
re-executing said handler method to report and/or record information about said first method; and,
executing a remaining portion of said first method through said exit point.

29. (Canceled).

30. (Currently Amended) The article of manufacture of ~~claim1-8~~ claim 18 wherein g) further comprises also identifying a second plug-in module for said third method based upon said third method and second classfile identification, said second plug-in module containing a second handler method.

31. (Previously Presented) The article of manufacture of claim 30 further comprising also executing said second handler method to report and/or record different information about said fourth method than what said first handler method reports and/or records about said fourth method.

32. (Previously Presented) The article of manufacture of claim 31 wherein a first object is called to execute said first method and a second object is called to execute said third method.

33. (Previously Presented) The article of manufacture of claim 32 wherein said object oriented run-time environment is a Java object oriented environment.

34. (Previously Presented) The article of manufacture of claim 18 wherein said invoking at a) above further comprises providing said first method's signature, said first method's signature comprising:
said identification of said first method;
said identification of said classfile that said first method is a part of; and,
said first method's arguments.

35. - 43. (Canceled).